METHOD AND APPARATUS FOR SPEECH SYNTHESIS WITHOUT PROSODY MODIFICATION

REFERENCE TO RELATED APPLICATION

The present application is a continuation-in-part of and claims priority of U.S. Patent Application serial number 09/850,527, filed May 7, 2001, entitled "Method and Apparatus for Speech Synthesis without Prosody Modification", which is based on and claims the benefit of U.S. Provisional application having serial number 60/251,167, filed on December 4, 2000 and entitled "PROSODIC WORD SEGMENTATION AND MULTI-TIER NON-UNIFORM UNIT SELECTION".

BACKGROUND OF THE INVENTION

The present invention relates to speech synthesis. In particular, the present invention relates to prosody in speech synthesis.

Text-to-speech technology allows computerized systems to communicate with users through synthesized speech. The quality of these systems is typically measured by how natural or human-like the synthesized speech sounds.

Very natural sounding speech can be produced by simply replaying a recording of an entire sentence or paragraph of speech. However, the complexity of human languages and the limitations of computer storage may make it impossible to store every conceivable sentence that may occur in a text. Because of this, the art has adopted a concatenative approach to speech synthesis that can be used to

generate speech from any text. This concatenative approach combines stored speech samples representing small speech units such as phonemes, diphones, triphones, or syllables to form a larger speech signal.

One problem with such concatenative systems is that a stored speech sample has a pitch and duration that is set by the context in which the sample was spoken. For example, in the sentence "Joe went to the store" the speech units associated with the word "store" have a lower pitch than in the question "Joe went to the store?" Because of this, if stored samples are simply retrieved without reference to their pitch or duration, some of the samples will have the wrong pitch and/or duration for the sentence resulting in unnatural sounding speech.

One technique for overcoming this is to identify the proper pitch and duration for each sample. Based on this prosody information, a particular sample may be selected and/or modified to match the target pitch and duration.

Identifying the proper pitch and duration is known as prosody prediction. Typically, it involves generating a model that describes the most likely pitch and duration for each speech unit given some text. The result of this prediction is a set of numerical targets for the pitch and duration of each speech segment.

These targets can then be used to select and/or modify a stored speech segment. For example, the targets can be used to first select the speech

segment that has the closest pitch and duration to the target pitch and duration. This segment can then be used directly or can be further modified to better match the target values.

For example, one prior art technique for modifying the prosody of speech segments is the so-called Time-Domain Pitch-Synchronous Overlap-and-Add (TD-PSOLA) technique, which is described in "Pitch-Synchronous Waveform Processing Techniques for Text-to-Speech Synthesis using Diphones", E. Moulines and F. Charpentier, Speech Communication, vol. 9, no. 5, pp. 453-467, 1990. Using this technique, the prior art increases the pitch of a speech segment by identifying a section of the speech segment responsible for the pitch. This section is a complex waveform that is a sum of sinusoids at multiples of a fundamental frequency F_0. The pitch period is defined by the distance between two pitch peaks in the waveform.

To increase the pitch, the prior art copies a segment of the complex waveform that is as long as the pitch period. This copied segment is then shifted by some portion of the pitch period and reinserted into the waveform. For example, to double the pitch, the copied segment would be shifted by one-half the pitch period, thereby inserting a new peak half-way between two existing peaks and cutting the pitch period in half.

To lengthen a speech segment, the prior art copies a section of the speech segment and inserts the copy into the complex waveform. In other words,

the entire portion of the speech segment after the copied segment is time-shifted by the length of the copied section so that the duration of the speech unit increases.

Unfortunately, these techniques for modifying the prosody of a speech unit have not produced completely satisfactory results. In particular, these modification techniques tend to produce mechanical or "buzzy" sounding speech.

Thus, it would be desirable to be able to select a stored unit that provides good prosody without modification. However, because of memory limitations, samples cannot be stored for all of the possible prosodic contexts in which a speech unit may be used. Instead, a limited set of samples must be selected for storage. Because of this, the performance of a system that uses stored samples without prosody modification is dependent on what samples are stored.

Thus, there is an ongoing need for improving the selection of these stored samples in systems that do not modify the prosody of the stored samples. There is also an ongoing need to reduce the computational complexity associated with identifying the proper prosody for the speech units.

SUMMARY OF THE INVENTION

A speech synthesizer is provided that concatenates stored samples of speech units without modifying the prosody of the samples. The present invention is able to achieve a high level of naturalness in synthesized speech with a carefully

designed speech corpus by storing samples based on the prosodic and phonetic context in which they occur. In particular, some embodiments of the present invention limit the training text to those sentences that will produce the most frequent sets of prosodic contexts for each speech unit. Further embodiments of the present invention also provide a multi-tier selection mechanism for selecting a set of samples that will produce the most natural sounding speech.

Under those embodiments that limit the training text, only a limited set of the sentences in a very large corpus are selected and read by a human into a training speech corpus from which samples of units are selected to produce natural sounding speech. To identify which sentences are to be read, embodiments of the present invention determine a frequency of occurrence for each context vector associated with a speech unit. Context vectors with a frequency of occurrence that is larger than a certain threshold are identified as necessary context vectors. Sentences that include the most necessary context vectors are selected for recording until all of the necessary context vectors have been included in the selected sub-set of sentences.

In embodiments that use a multi-tier selection method, a set of candidate speech segments is identified for each speech unit by comparing the input context vector to the context vectors associated with the speech segments. A path through the candidate speech segments is then selected based

on differences between the input context vectors and the stored context vectors as well as some cost function that indicates the prosodic smoothness of the resulting concatenated speech signal. Under one embodiment, the cost function gives preference to selecting a series of speech segments that appeared next to each other in the training corpus. In further embodiments, the cost function cost function comprises linear, and/or, one or more higher order coordinates being combinations of at least two factors from a set of factors. The set of factors include: an indication of a position of a speech unit in a phrase; an indication of a position of a speech unit in a word; an indication of a category for a phoneme preceding a speech unit; an indication of a category for a phoneme following a speech unit; an indication of a category for tonal identity of the current speech unit; an indication of a category for tonal identity of a preceding speech unit; an indication of a category for tonal identity of a following speech unit; an indication of a level of stress of a speech unit; an indication of a coupling degree of pitch, duration and/or energy with a neighboring unit; and an indication of a degree of spectral mismatch with a neighboring speech unit. A technique for obtaining an optimized cost function, which can be used in a speech synthesizer as the criterion for the unit selection, is also provided below.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of a general computing environment in which the present invention may be practiced.

FIG. 2 is a block diagram of a mobile device in which the present invention may be practiced.

FIG. 3 is a block diagram of a speech synthesis system.

FIG. 4 is a block diagram of a system for selecting a training text subset from a very large training corpus.

FIG. 5 is a flow diagram for constructing a decision tree under one embodiment of the present invention.

FIG. 6 is a block diagram of a multi-tier selection system for selecting speech segments under embodiments of the present invention.

FIG. 7 is a flow diagram of a multi-tier selection system for selecting speech segments under embodiments of the present invention.

FIG. 8 is a flow diagram for estimating mean opinion score from a context vector or an objective measure.

FIG. 9 is a plot of a relationship between mean opinion score and the objective measure.

FIG. 10 is a flow diagram illustrating a method for optimizing the objective measure.

FIG. 11 is a flow diagram illustrating an exemplary method of optimizing the objective measure.

DETAILED DESCRIPTION OF ILLUSTRATIVE EMBODIMENTS

FIG. 1 illustrates an example of a suitable computing system environment 100 on which the invention may be implemented. The computing system environment 100 is only one example of a suitable computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the invention. Neither should the computing environment 100 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary operating environment 100.

The invention is operational with numerous other general purpose or special purpose computing system environments or configurations. Examples of well known computing systems, environments, and/or configurations that may be suitable for use with the invention include, but are not limited to, personal computers, server computers, hand-held or laptop devices, multiprocessor systems, microprocessor-based systems, set top boxes, programmable consumer electronics, network PCs, minicomputers, mainframe computers, distributed computing environments that include any of the above systems or devices, and the like.

The invention may be described in the general context of computer-executable instructions, such as program modules, being executed by a computer. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or

implement particular abstract data types. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote computer storage media including memory storage devices. Tasks performed by the programs and modules are described below and with the aid of figures. Those skilled in the art can implement the description and figures as processor executable instructions, which can be written on any form of a computer readable media.

With reference to FIG. 1, an exemplary system for implementing the invention includes a general-purpose computing device in the form of a computer 110. Components of computer 110 may include, but are not limited to, a processing unit 120, a system memory 130, and a system bus 121 that couples various system components including the system memory to the processing unit 120. The system bus 121 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. By way of example, and not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnect (PCI) bus also known as Mezzanine bus.

Computer 110 typically includes a variety of computer readable media. Computer readable media can be any available media that can be accessed by computer 110 and includes both volatile and nonvolatile media, removable and non-removable media. By way of example, and not limitation, computer readable media may comprise computer storage media and communication media. Computer storage media includes both volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by computer 100.

Communication media typically embodies computer readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired

media such as a wired network or direct-wired connection, and wireless media such as acoustic, FR, infrared and other wireless media. Combinations of any of the above should also be included within the scope of computer readable media.

The system memory 130 includes computer storage media in the form of volatile and/or nonvolatile memory such as read only memory (ROM) 131 and random access memory (RAM) 132. A basic input/output system 133 (BIOS), containing the basic routines that help to transfer information between elements within computer 110, such as during start-up, is typically stored in ROM 131. RAM 132 typically contains data and/or program modules that are immediately accessible to and/or presently being operated on by processing unit 120. By way of example, and not limitation, FIG. 1 illustrates operating system 134, application programs 135, other program modules 136, and program data 137.

The computer 110 may also include other removable/non-removable volatile/nonvolatile computer storage media. By way of example only, FIG. 1 illustrates a hard disk drive 141 that reads from or writes to non-removable, nonvolatile magnetic media, a magnetic disk drive 151 that reads from or writes to a removable, nonvolatile magnetic disk 152, and an optical disk drive 155 that reads from or writes to a removable, nonvolatile optical disk 156 such as a CD ROM or other optical media. Other removable/non-removable, volatile/nonvolatile computer storage media that can be used in the exemplary operating

environment include, but are not limited to, magnetic tape cassettes, flash memory cards, digital versatile disks, digital video tape, solid state RAM, solid state ROM, and the like. The hard disk drive 141 is typically connected to the system bus 121 through a non-removable memory interface such as interface 140, and magnetic disk drive 151 and optical disk drive 155 are typically connected to the system bus 121 by a removable memory interface, such as interface 150.

The drives and their associated computer storage media discussed above and illustrated in FIG. 1, provide storage of computer readable instructions, data structures, program modules and other data for the computer 110. In FIG. 1, for example, hard disk drive 141 is illustrated as storing operating system 144, application programs 145, other program modules 146, and program data 147. Note that these components can either be the same as or different from operating system 134, application programs 135, other program modules 136, and program data 137. Operating system 144, application programs 145, other program modules 146, and program data 147 are given different numbers here to illustrate that, at a minimum, they are different copies.

A user may enter commands and information into the computer 110 through input devices such as a keyboard 162, a microphone 163, and a pointing device 161, such as a mouse, trackball or touch pad. Other input devices (not shown) may include a joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to

the processing unit 120 through a user input interface 160 that is coupled to the system bus, but may be connected by other interface and bus structures, such as a parallel port, game port or a universal serial bus (USB). A monitor 191 or other type of display device is also connected to the system bus 121 via an interface, such as a video interface 190. In addition to the monitor, computers may also include other peripheral output devices such as speakers 197 and printer 196, which may be connected through an output peripheral interface 190.

The computer 110 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 180. The remote computer 180 may be a personal computer, a hand-held device, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to the computer 110. The logical connections depicted in FIG. 1 include a local area network (LAN) 171 and a wide area network (WAN) 173, but may also include other networks. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

When used in a LAN networking environment, the computer 110 is connected to the LAN 171 through a network interface or adapter 170. When used in a WAN networking environment, the computer 110 typically includes a modem 172 or other means for establishing communications over the WAN 173, such as

the Internet. The modem 172, which may be internal or external, may be connected to the system bus 121 via the user input interface 160, or other appropriate mechanism. In a networked environment, program modules depicted relative to the computer 110, or portions thereof, may be stored in the remote memory storage device. By way of example, and not limitation, FIG. 1 illustrates remote application programs 185 as residing on remote computer 180. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

To further help understand the usefulness of the present invention, it may helpful to provide a brief description of a speech synthesizer 300 illustrated in FIG. 3. However, it should be noted that the synthesizer 300 is provided for exemplary purposes and is not intended to limit the present invention.

FIG. 2 is a block diagram of a mobile device 200, whichis an exemplary computing environment. Mobile device 200 includes a microprocessor 202, memory 204, input/output (I/O) components 206, and acommunication interface 208 for communicating with remote computers or other mobile devices. In one embodiment, the afore-mentioned components are coupled for communication with one another over a suitable bus 210.

Memory 204 is implemented as non-volatile electronic memory such as random access memory (RAM)

with a battery back-up module (not shown) such that information stored in memory 204 is not lost when the general power to mobile device 200 is shut down. A portion of memory 204 is preferably allocated as addressable memory for program execution, while another portion of memory 204 is preferably used for storage, such as to simulate storage on a disk drive.

Memory 204 includes an operating system 212, application programs 214 as well as an object store 216. During operation, operating system 212 is preferably executed by processor 202 from memory 204. Operating system 212, in one preferred embodiment, is a WINDOWS® CE brand operating system commercially available from Microsoft Corporation. Operating system 212 is preferably designed for mobile devices, and implements database features that can be utilized by applications 214 through a set of exposed application programming interfaces and methods. The objects in object store 216 are maintained by applications 214 and operating system 212, at least partially in response to calls to the exposed application programming interfaces and methods.

Communication interface 208 represents numerous devices and technologies that allow mobile device 200 to send and receive information. The devices include wired and wireless modems, satellite receivers and broadcast tuners to name a few. Mobile device 200 can also be directly connected to a computer to exchange data therewith. In such cases, communication interface 208 can be an infrared transceiver or a serial or parallel communication

connection, all of which are capable of transmitting streaming information.

Input/output components 206 include a variety of input devices such as a touch-sensitive screen, buttons, rollers, and a microphone as well as a variety of output devices including an audio generator, a vibrating device, and a display. The devices listed above are by way of example and need not all be present on mobile device 200. In addition, other input/output devices may be attached to or found with mobile device 200 within the scope of the present invention.

Under the present invention, a speech synthesizer is provided that concatenates stored samples of speech units without modifying the prosody of the samples. The present invention is able to achieve a high level of naturalness in synthesized speech with a carefully designed speech corpus by storing samples based on the prosodic and phonetic context in which they occur. In particular, the present invention limits the training text to those sentences that will produce the most frequent sets of prosodic contexts for each speech unit. The present invention also provides a multi-tier selection mechanism for selecting a set of samples that will produce the most natural sounding speech.

FIG. 3 is a block diagram of a speech synthesizer 300 that is capable of constructing synthesized speech 302 from an input text 304 under embodiments of the present invention. In conventional concatenative TTS systems, a pitch and duration

modification algorithm, such as PSOLA, is applied to pre-stored units to guarantee that the prosodic features of synthetic speech meet the predicted target values. These systems have the advantages of flexibility in controlling the prosody. Yet, they often suffer from significant quality decrease in naturalness. In the TTS system 300, speech is generated by directly concatenating speech segments (for speech units such as syllables, phonemes, diphones, semiphones, etc.) without any pitch or duration modification under the assumption that the speech database contains enough prosodic and spectral varieties for all speech units and the best fitting segments can always be found.

Before speech synthesizer 300 can be utilized to construct speech 302, it must be initialized with samples of speech units taken from a training text 306 that is read into speech synthesizer 300 as training speech 308.

As noted above, speech synthesizers are constrained by a limited size memory. Because of this, training text 306 must be limited in size to fit within the memory. However, if the training text is too small, there will not be enough samples of the training speech to allow for concatenative synthesis without prosody modifications. One aspect of the present invention overcomes this problem by trying to identify a set of speech units in a very large text corpus that must be included in the training text to allow for concatenative synthesis without prosody modifications.

FIG. 4 provides a block diagram of components used to identify smaller training text 306 of FIG. 3 from a very large corpus 400. Under one embodiment, very large corpus 400 is a corpus of five years worth of the People's Daily, a Chinese newspaper, and contains about 97 million Chinese Characters.

Initially, training text 400 is parsed by a parser/semantic identifier 402 into strings of individual speech units attached with various textual information. Under some embodiments of the invention, especially those used to form Chinese speech, the speech units are tonal syllables. However, other speech units such as phonemes, diphones, triphones or the mix of them may be used within the scope of the present invention.

Parser/semantic identifier 402 also identifies high-level prosodic information about each sentence provided to the parser 402. This high-level prosodic information includes the predicted tonal levels for each speech unit as well as the grouping of speech units into prosodic words and phrases. In embodiments where tonal syllable speech units are used, parser/semantic identifier 402 also identifies the first and last phoneme in each speech unit.

The strings of speech units attached with textual and prosodic information produced from the training text are provided to a context vector generator 404, which generates a Speech-unit Dependent Descriptive Contextual Variation Vector (SDDCVV, hereinafter referred to as a "context

vector"). The context vector describes several context variables that can affect the naturalness of the speech unit. Under one embodiment, the context vector describes six variables or coordinates of textual information. They are:

Position in phrase (PinP): the position of the current speech unit in its carrying prosodic phrase.

Position in word (PinW): the position of the current speech unit in its carrying prosodic word.

Left phonetic context (LPhC): category of the last phoneme in the speech unit to the left (preceding) of the current speech unit.

Right phonetic context (RPhC): category of the first phoneme in the speech unit to the right (following) of the current speech unit.

Left tone context (LTC): the tone category of the speech unit to the left (preceding) of the current speech unit.

Right tone context (RTC): the tone category of the speech unit to the right (following) of the current speech unit.

If desired, the coordinates of the context vector can also include the stress level of the current speech unit, the tonal identity of current speech unit or

the coupling degree of its pitch, duration and/or energy with its neighboring units.

Under one embodiment, the position in phrase coordinate and the position in word coordinate can each have one of four values, the left phonetic context can have one of eleven values, the right phonetic context can have one of twenty-six values and the left and right tonal contexts can each have one of two values. Under this embodiment, there are 4*4*11*26*2*2= 18304 possible context vectors for each speech unit.

The context vectors produced by generator 404 are grouped based on their speech unit. For each speech unit, a frequency-based sorter 406 identifies the most frequent context vectors for each speech unit. The most frequently occurring context vectors for each speech unit are then stored in a list of necessary context vectors 408. In one embodiment, the top context vectors, whose accumulated frequency of occurrence is not less than half of the total frequency of occurrence of all units, are stored in the list.

The sorting and pruning performed by sorter 406 is based on a discovery made by the present inventors. In particular, the present inventors have found that certain context vectors occur repeatedly in the corpus. By making sure that these context vectors are found in the training corpus, the present invention increases the chances of having an exact context match for an input text without greatly increasing the size of the training corpus. For

example, the present inventors have found that by ensuring that the top two percent of the context vectors are represented in the training corpus, an exact context match will be found for an input text speech unit over fifty percent of the time.

Using the list of necessary context vectors 408, a text selection unit 410 selects sentences from very large corpus 400 to produce training text subset 306. In a particular embodiment, text selection unit 410 uses a greedy algorithm to select sentences from corpus 400. Under this greedy algorithm, selection unit 410 scans all sentences in the corpus and picks out one at a time to add to the selected group.

During the scan, selection unit 410 determines how many context vectors in list 408 are found in each sentence. The sentence that contains the maximum number of needed context vectors is then added to training text 306. The context vectors that the sentence contains are removed from list 408 and the sentence is removed from the large text corpus 400. The scanning is repeated until all of the context vectors have been removed from list 408.

After training text subset 306 has been formed, it is read by a person and digitized into a training speech corpus. Both the training text and training speech can be used to initialize speech synthesizer 300 of FIG. 3. This initialization begins by parsing the sentences of text 306 into individual speech units that are annotated with high-level prosodic information. In FIG. 3, this is accomplished by a parser/semantic identifier 310,

which is similar to parser/semantic identifier 402 of FIG. 4. The parsed speech units and their high-level prosodic description are then provided to a context vector generator 312, which is similar to context vector generator 404 of FIG. 4.

The context vectors produced by context vector generator 312 are provided to a component storing unit 314 along with speech samples produced by a sampler 316 from training speech signal 308. Each sample provided by sampler 316 corresponds to a speech unit identified by parser 310. Component storing unit 314 indexes each speech sample by its context vector to form an indexed set of stored speech components 318.

Under one embodiment, the samples are indexed, for example, by a prosody-dependent decision tree (PDDT), which is formed automatically using a classification and regression tree (CART). CART provides a mechanism for selecting questions that can be used to divide the stored speech components into small groups of similar speech samples. Typically, each question is used to divide a group of speech components into two smaller groups. With each question, the components in the smaller groups become more homogenous. The process for using CART to form the decision tree is shown in FIG. 5.

At step 500 of FIG. 5, a list of candidate questions is generated for the decision tree. Under one embodiment, each question is directed toward some coordinate or combination of coordinates in the context vector.

At step 502, an expected square error is determined for all of the training samples from sampler 316. The expected square error gives a measure of the distances among a set of features of each sample in a group. In one particular embodiment, the features are prosodic features of average fundamental frequency (F_a), average duration (F_b), and range of the fundamental frequency (F_c) for a unit. For this embodiment, the expected square error is defined as:

$$ESE(t) = E\left(W_a E_a + W_b E_b + W_c E_c\right) \qquad \text{EQ. 1}$$

where ESE(t) is the expected square error for all samples X on node t in the decision tree, E_a, E_b, and E_c are the square error for F_a, F_b, and F_c, respectively, W_a, W_b, and W_c are weights, and the operation of determining the expected value of the sum of square errors is indicated by the outer E().

Each square error is then determined as:

$$E_j = \left|F_j - R\left(F_j\right)\right|^2 \quad , j = a,b,c \qquad \text{EQ. 2}$$

where $R(F_j)$ is a regression value calculated from samples X on node t. In this embodiment, the regression value is the expected value of the feature as calculated from the samples X at node t: $R_j(F_j) = E(F_j / X \in node_t)$.

Once the expected square error has been determined at step 502, the first question in the question list is selected at step 504. The selected question is applied to the context vectors at step 506 to group the samples into candidate sub-nodes for

the tree. The expected square error of each sub-node is then determined at step 508 using equations 1 and 2 above.

At step 510, a reduction in expected square error created by generating the two sub-nodes is determined. Under one embodiment, this reduction is calculated as:

$$\Delta WESE(t) = ESE(t)P(t) - (ESE(l)P(l) + ESE(r)P(r)) \quad \text{EQ. 3}$$

where $\Delta WESE(t)$ is the reduction in expected square error, ESE(t) is the expected square error of node t, against which the question was applied, P(t) is the percentage of samples in node t, ESE(l) and ESE(r) are the expected square error of the left and right sub-nodes formed by the question, respectively, and P(l) and P(r) are the percentage of samples in the left and right node, respectively.

The reduction in expected square error provided by the current question is stored and the CART process determines if the current question is the last question in the list at step 512. If there are more questions in the list, the next question is selected at step 514 and the process returns to step 506 to divide the current node into sub-nodes based on the new question.

After every question has been applied to the current node at step 512, the reductions in expected square error provided by each question are compared and the question that provides the greatest reduction is set as the question for the current node of the decision tree at step 515.

At step 516, a decision is made as to whether or not the current set of leaf nodes should be further divided. This determination can be made based on the number of samples in each leaf node or the size of the reduction in square error possible with further division.

Under one embodiment, when the decision tree is in its final form, each leaf node will contain a number of samples for a speech unit. These samples have slightly different prosody from each other. For example, they may have different phonetic contexts or different tonal contexts from each other. By maintaining these minor differences within a leaf node, this embodiment of the invention introduces slender diversity in prosody, which is helpful in removing monotonous prosody.

If the current leaf nodes are to be further divided at step 516, a leaf node is selected at step 518 and the process returns to step 504 to find a question to associate with the selected node. If the decision tree is complete at step 516, the process of FIG. 5 ends at step 520.

The process of FIG. 5 results in a prosody-dependent decision tree 320 of FIG. 3 and a set of stored speech samples 318, indexed by decision tree 320. Once created, decision tree 320 and speech samples 318 can be used under further aspects of the present invention to generate concatenative speech without requiring prosody modification.

The process for forming concatenative speech begins by parsing a sentence in input text 304

using parser/semantic identifier 310 and identifying high-level prosodic information for each speech unit produced by the parse. This prosodic information is then provided to context vector generator 312, which generates a context vector for each speech unit identified in the parse. The parsing and the production of the context vectors are performed in the same manner as was done during the training of prosody decision tree 320.

The context vectors are provided to a component locator 322, which uses the vectors to identify a set of samples for the sentence. Under one embodiment, component locator 322 uses a multi-tier non-uniform unit selection algorithm to identify the samples from the context vectors.

FIGS. 6 and 7 provide a block diagram and a flow diagram for the multi-tier non-uniform selection algorithm. In step 700, each vector in the set of input context vectors is applied to prosody-dependent decision tree 320 to identify a leaf node array 600 that contains a leaf node for each context vector. At step 702, a set of distances is determined by a distance calculator 602 for each input context vector. In particular, a separate distance is calculated between the input context vector and each context vector found in its respective leaf node. Under one embodiment, each distance is calculated as:

$$D_c = \sum_{i=1}^{I} W_{ci} D_i \qquad \text{EQ. 4}$$

where D_c is the context distance, D_i is the distance for coordinate i of the context vector, W_{ci} is a

weight associated with coordinate i, and I is the number of coordinates in each context vector.

At step 704, the N samples with the closest context vectors are retained while the remaining samples are pruned from node array 600 to form pruned leaf node array 604. The number of samples, N, to leave in the pruned nodes is determined by balancing improvements in prosody with improved processing time. In general, more samples left in the pruned nodes means better prosody at the cost of longer processing time.

At step 706, the pruned array is provided to a Viterbi decoder 606, which identifies a lowest cost path through the pruned array. Under a single-tier embodiment of the present invention, the lowest cost path is identified simply by selecting the sample with the closest context vector in each node. Under a multi-tier embodiment, the cost function is modified to be:

$$C_c = W_c \sum_{j=1}^{J} D_{cj} + W_s \sum_{j=1}^{J} C_{sj} \qquad \text{EQ. 5}$$

where C_c is the concatenation cost for the entire sentence, W_c is a weight associated with the distance measure of the concatenated cost, D_{cj} is the distance calculated in equation 4 for the j^{th} speech unit in the sentence, W_s is a weight associated with a smoothness measure of the concatenated cost, C_{sj} is a smoothness cost for the j^{th} speech unit, and J is the number of speech units in the sentence.

The smoothness cost in Equation 5 is defined to provide an objective measure of the

prosodic mismatch between sample j and the samples proposed as the neighbors to sample j by the Viterbi decoder. Under one embodiment, the smoothness cost is determined based on whether a sample and its neighbors were found as neighbors in an utterance in the training corpus. If a sample occurred next to its neighbors in the training corpus, the smoothness cost is zero since the samples contain the proper spectral transition in between. If a sample did not occur next to its neighbors in the training corpus (referred as non-neighboring case), the smoothness cost is set to one. Under another embodiment, different values are assigned to the smoothness cost for non-neighboring cases according to their boundary types. For example, if the boundary between the two segments is sonorant to sonorant, the largest cost (1) is given. If the boundary between them is non-sonorant consonant to non-sonorant consonant, a small cost (0.2) is given. The cost between sonorant to non-sonorant or non-sonorant to sonorant transition is in middle (0.5). The different smoothness costs lead the search algorithm to prefer concatenation at boundaries with smaller cost.

Using the multi-tier non-uniform approach, if a large block of speech units, such as a word or a phrase, in the input text exists in the training corpus, preference will be given to selecting all of the samples associated with that block of speech units. Note, however, that if the block of speech units occurred within a different prosodic context, the distance between the context vectors will likely

cause different samples to be selected than those associated with the block.

Once the lowest cost path has been identified by Viterbi decoder 606, the identified samples 608 are provided to speech constructor 303. With the exception of small amounts of smoothing at the boundaries between the speech units, speech constructor 303 simply concatenates the speech units to form synthesized speech 302. Thus, the speech units are combined without having to change their prosody.

The cost function or objective measure provided above contains only first order components of the seven textual factors, yet, higher order interactions might exist among these factors. In further embodiments, the context vector, cost function or objective measure comprises one or more higher order coordinates being combinations of at least two factors from a set of factors including: an indication of a position of a speech unit in a phrase; an indication of a position of a speech unit in a word; an indication of a category for a phoneme preceding a speech unit; an indication of a category for a phoneme following a speech unit; an indication of a category for tonal identity of the current speech unit; an indication of a category for tonal identity of a preceding speech unit; an indication of a category for tonal identity of a following speech unit; an indication of a level of stress of a speech unit; an indication of a coupling degree of pitch,

duration and/or energy with a neighboring unit; and an indication of a degree of spectral mismatch with a neighboring speech unit. A technique for obtaining an optimized cost function, which can be used in the speech synthesizer 300 as the criterion for the unit selection, is also provided below.

Evaluating the quality of synthesized speech contains two aspects, intelligibility and naturalness. Generally, intelligibility is not a large concern for most text-to-speech systems. However, the naturalness of synthesized speech is a larger issue and is still far from most expectations.

During text-to-speech system development, it is necessary to have regular evaluations on a naturalness of the system. The Mean Opinion Score (MOS) is one of the most popular and widely accepted subjective measures for naturalness. However, running a formal MOS evaluation is expensive and time consuming. Generally, to obtain a MOS score for a system under consideration, a collection of synthesized waveforms must be obtained from the system. The synthesized waveforms, together with some waveforms generated from other text-to-speech systems and/or waveforms uttered by a professional announcer are randomly played to a set of subjects. Each of the subjects are asked to score the naturalness of each waveform from 1-5 (1=bad, 2=poor, 3=fair, 4=good, 5=excellent). The means of the scores from the set of subjects for a given waveform represents naturalness in a MOS evaluation. Recently, a method for

estimating mean opinion score or naturalness of synthesized speech has been advanced by Chu, M. and Peng, H., in "An objective measure for estimating MOS of synthesized speech", *Proceedings of Eurospeech2001*, 2001, and is discussed further below. The method includes using an objective measure that has components derived directly from textual information used to form synthesized utterances. The objective measure has a high correlation with the mean opinion score such that a relationship can be formed between the objective measure and the corresponding mean opinion score. An estimated mean opinion score can be obtained easily from the relationship when the objective measure is applied to utterances of a modified speech synthesizer.

The objective measure can be based on one or more factors of the speech units used to create the utterances. The factors can include the position of the speech unit in a phrase or word, the neighboring phonetic or tonal context, the spectral mismatch of successive speech units or the stress level of the speech unit. Weighting factors can be used since correlation of the factors with mean opinion score has been found to vary between the factors.

By using the objective measure it is easy to track performance in naturalness of the speech synthesizer, thereby allowing efficient development of the speech synthesizer. In particular, the objective measure can serve as criteria for

optimizing the algorithms for speech unit selection and speech database pruning.

As discussed above, the evaluation of concatenative cost can form the basis of an objective measure for MOS estimation. A method for using the objective measure in estimating MOS is illustrated in FIG. 8. Generally, the method includes generating a set of synthesized utterances at step 800, and subjectively rating each of the utterances at step 802. A score is then calculated for each of the synthesized utterances using the objective measure at step 804. The scores from the objective measure and the ratings from the subjective analysis are then analyzed to determine a relationship at step 806. The relationship is used at step 808 to estimate naturalness or MOS when the objective measure is applied to the textual information of speech units for another utterance or second set of utterances from a modified speech synthesizer (e.g. when a parameter of the speech synthesizer has been changed). It should be noted that the words of the "another utterance" or the "second set of utterances" obtained from the modified speech synthesizer can be the same or different words used in the first set of utterances.

In one embodiment, in order to make the concatenative cost comparable among utterances with variable number of syllables, the average concatenative cost of an utterance is used and can be expressed as:

$$C_a = \sum_{i=1}^{I+1} W_i C_{ai} \qquad EQ.\ 6$$

$$C_{ai} = \begin{cases} \frac{1}{J}\sum_{l=1}^{J} D_i(l), & i = 1,\cdots,I \\ \frac{1}{J-1}\sum_{l=1}^{J-1} C_s(l), & i = I+1 \end{cases}$$

$$W_i = \begin{cases} W_{ci} W_c & i = 1,\cdots,I \\ W_s & i = I+1 \end{cases}$$

where, C_a is the average concatenative cost and C_{ai} (i=1,...,7) one or more of the factors that contribute to C_a, which are, in the illustrative embodiment, the average costs for position in phrase ("PinP"), position in word ("PinW"), left phonetic context ("LPhC"), right phonetic context ("RPhC"), left tone context ("LTC"), right tone context ("RTC") and smoothness cost per unit in an utterance.

The cost function as provided above is a weighted sum of seven factors. Six of the factors are distances between the target category and the category of candidate unit (named as unit category) for the six contextual factors, which are PinP, PinW, LPhC, RPhC, LTC and RTC. Since all these factors take only categorical values, the distance between categories are empirically predefined in distance tables. The seventh factor is an enumerated smoothness cost, which takes value 0 when current candidate unit is a continuous segment with the unit before it in the unit inventory and takes value larger than 0 otherwise.

W_i are weights for the seven component-costs and all are set to 1, but can be changed. For instance, it has been found that the coordinate having the highest correlation with mean opinion score was smoothness, whereas the lowest correlation with mean opinion score was position in phase. It is therefore reasonable to assign larger weights for components with high correlation and smaller weights for components with low correlation. In one experiment, the following weights were used:

Position in Phrase, $W_1 = 0.10$
Position in Word, $W_2 = 0.60$
Left Phonetic Context, $W_3 = 0.10$
Right Phonetic Context, $W_4 = 0.76$
Left Tone Context, $W_5 = 1.76$
Right Tone Context, $W_6 = 0.72$
Smoothness, $W_7 = 2.96$

In one exemplary embodiment, 100 sentences are carefully selected from a 200 MB text corpus so the C_a and C_{ai} (i=1,...,7) of them are scattered into wide spans. Four synthesized waveforms are generated for each sentence with the speech synthesizer 200 above with four speech databases, whose sizes are 1.36 GB, 0.9 GB, 0.38 GB and 0.1 GB, respectively. C_a and C_{ai} of each waveform are calculated. All the 400 synthesized waveforms, together with some waveforms generated from other TTS systems and waveforms uttered by a professional announcer, are randomly played to 30 subjects. Each of the subjects is asked to score the naturalness of each waveform from 1-5

(1=bad, 2=poor, 3=fair, 4=good, 5=excellent). The mean of the thirty scores for a given waveform represents its naturalness in MOS.

Fifty original waveforms uttered by the speaker who provides voice for the speech database are used in this example. The average MOS for these waveforms was 4.54, which provides an upper bound for MOS of synthetic voice. Providing subjects a wide range of speech quality by adding waveforms from other systems can be helpful so that the subjects make good judgements on naturalness. However, only the MOS for the 400 waveforms generated by the speech synthesizer under evaluation are used in conjunction with the corresponding average concatenative cost score.

FIG. 9 is a plot illustrating the objective measure (average concatenative cost) versus subjective measure (MOS) for the 400 waveforms. A correlation coefficient between the two dimensions is -0.822, which reveals that the average concatenative cost function replicates, to a great extent, the perceptual behavior of human beings. The minus sign of the coefficient means that the two dimensions are negatively correlated. The larger C_a is, the smaller the corresponding MOS will be. A linear regression trendline 902 is illustrated in FIG. 9 and is estimated by calculating the least squares fit throughout points. The trendline or curve is denoted as the average concatenative cost-MOS curve and for the exemplary embodiment is:

$Y=-1.0327x + 4.0317.$

However, it should be noted that analysis of the relationship of average concatenative cost and MOS score for the representative waveforms can also be performed with other curve-fitting techniques, using, for example, higher-order polynomial functions. Likewise, other techniques of correlating average concatenative cost and MOS can be used. For instance, neural networks and decision trees can also be used.

Using the average concatenative cost vs. MOS relationship, an estimate of MOS for a single synthesized speech waveform can be obtained by its average concatenative cost. Likewise, an estimate of the average MOS for a TTS system can be obtained from the average of the average of the concatenative costs that are calculated over a large amount of synthesized speech waveforms. In fact, when calculating the average concatenative cost, it is unnecessary to generate the speech waveforms since the costs can be calculated after the speech units have been selected.

Although the concatenative cost function has proven to replicate, to a great extent, the perceptual behavior of human beings, it might not be optimal. It has been discovered by the inventors that some factors that can contribute to inaccuracies in the concatenative cost function include that many parameters in the cost function are assigned empirically by a human expert, and accordingly, they might not be the most suitable values. In addition, the concatenative cost function provided above

contains only first order components of the seven textual factors, yet, higher order interactions might exist among these factors. Furthermore, there might be other components that could be added into the concatenative cost function.

A method for optimizing the objective measure or concatenative cost function for unit selection in the corpus-based TTS system by maximizing the correlation between the concatenative cost and the MOS is provided. The method is illustrated in FIG. 10 at 1000. At step 1002, a subjective evaluation should be done first as discussed above. However, a beneficial aspect of this step is to log or record in a file or other means the textual information of all units appearing in the synthetic utterances evaluated. At step 1003, an initial concatenative cost function is used and a correlation with MOS is established. At step 1004, the concatenative cost function is altered, for example, using any one or more of the techniques described below. With the recorded log file, a new concatenative cost can be recalculated at step 1006 using the new a cost function. The correlation between the new concatenative cost and MOS is obtained at step 1008, which also serves as a measure for the validity of any change in the concatenative cost function. Steps 1004, 1006 and 1008 can be repeated as necessary until an optimized concatenative cost function is realized. It is important to note that only a single run of MOS evaluation (step 1002) is required in the

optimization method 1000. This is helpful because step 1002 can be particularly labor and time consuming. Other optimization algorithms such as Gradient Declination can also be used to optimize the free parameters.

As indicated above, in order to evaluate improvements and accuracy made to the cost function, a measure needs to be used. One useful measure has been found to be the correlation between the concatenative cost and the MOS such as illustrated in FIG. 9. Thus, if the correlation between concatenative cost and MOS improves with changes to the concatenative cost function, such changes can be included in the concatenative cost function.

As mentioned above, the log file of step 1002 keeps the information of the target units wanted and the units actually used. Concatenative cost for all sentences can be calculated with any new cost function from the log file. That is to say, the form of the cost function or the distance tables used by the cost function can be changed, and the validity of the change can be measured through movement of the correlation between the new cost and the MOS for the set of synthesized utterances. Furthermore, when a specific format is given to a cost function, the correlation between concatenative cost and MOS can be treated as a function of the parameters of the concatenative cost function, denoted by the following equation

$$Corr = f(x_1, x_2, \cdots, x_N) \qquad \text{EQ. 7}$$

where, N is number of free parameters in the concatenative cost function. If the concatenative cost function is defined as equation (3), distances between target and unit categories for the six textual factors and the weights for the seven factors can be free parameters. An optimization routine is used to optimize the free parameters so that the largest correlation is to be achieved. One suitable optimization routine that can be used is the function "fmincon" in the Matlab Optimization Toolbox by The MathWorks, Inc. of Natick, Massachusetts, U.S.A ("Optimization Toolbox User's Guide: For Use with MATLAB"), which searches for the minimum of a constrained nonlinear multivariable function, and optimizes the free parameters so that the largest correlation is to be achieved. Since concatenative cost and MOS is negatively correlated, Corr in equation 4 is to be minimized.

In one embodiment, for instance, depending on the number of utterances available with MOS scores, the number of free parameters in each run of optimization should not be too large. Thus, in one embodiment, optimization can be separated into many runs. In each run at step 1004, only some of the parameters are optimized and the others are fixed at their original values. Referring to FIG. 11, three different kinds of changes can be made to the concatenative cost function; specifically optimize the distance tables for the six single-order textual factors individually at step 1102; explore the interactions among factors and add some higher order

components into the cost function at step 1104; and optimize the weight for each component in the new cost function at step 1106.

Since some parameters in the distance tables may not be used frequently depending on the number of available sentences, optimizing them with a few observations will probably cause an overfitting problem. In this case, to avoid overfitting, a threshold can be set for the number of times a parameter had been used. Only frequently used ones are optimized. Though, no globally optimized solution is guaranteed, it is quite likely that the overall correlation is increased.

In order to check the validity of the optimized parameters, a K-fold cross validation experiment is done. In one embodiment, K is set to 4. In each run of optimization, only 300 utterances are used for training and the remaining 100 sentences are used for testing. If the difference between average correlation coefficients for the training and testing set is large, the optimization is considered invalid. Thus, the number of free parameters should be reduced. For valid optimization, means of the four sets of optimized parameters are used in the final cost function.

As indicated above, the distances between target categories and unit categories of a textual factor are assigned manually, which may not be the most suitable values. In a first method of optimization of the concatenative cost function, the distance table for each textual factor is improved at step 1102 individually. Here, the concatenative cost function contains only a single textual component in each run of optimization. The correlation coefficients between the six textual factors and MOS before and after optimization are listed in Table 1.

	IniCorr	TrCorr	TsCorr
PinP	0.498	0.525	0.498
PinW	0.623	0.631	0.623
LPhC	0.553	0.715	0.703
RPhC	0.688	0.742	0.736
LTC	0.654	0.743	0.731
RTC	0.622	0.755	0.732

Table 1

In Table 1, the correlation coefficients between the six textual factors and MOS before and after optimization are provided where "IniCorr" provides the initial coefficient obtained with the empirical distance tables; "TrCorr" provides the average coefficient on the four training sets after optimization; and "TsCorr" provides the average coefficient on testing sets after optimization.

It can be seen that there is no change for the correlation for the factor PinP and PinW on the testing set, and both of them have smaller correlation to MOS than other factors. The reason might be that both of them have been used in the splitting question for constructing indexing CART for the unit inventory. Thus, most of the units used in subjective experiment have zero distances for the two factors. For the other four factors, great increases are obtained.

Using the factor RTC by way of example for detailed explanation, the initial distance table and the optimized one for RTC are given in Table 2(a) and 2(b) below. T1 - T5 represent the four normal tones and the neutral tone in Mandarin Chinese. Rows in Tables 2(a) and 2(b) represent the target RTC, while the columns represent the unit RTC. The numbers in the tables are the distances between target RTC and unit RTC. It can be seen that many distances reach a more precise value after optimization, in comparison to those given by a human expert. There are some numbers unchanged in Table 2(b) since they haven't been used enough times in the training set. Thus, they are fixed at the initial values during the optimizing phase.

Target RTC / *Unit RTC*	*T1*	*T2*	*T3*	*T4*	*T5*
T1	*0*	*0.25*	*0.75*	*0.25*	*1*
T2	*0.5*	*0*	*0.25*	*0.75*	*1*
T3	*0.5*	*0.25*	*0*	*0.75*	*1*

T4	0.75	0.5	1	0	0.25
T5	0.5	0.75	1	0.25	0

Table 2(a) The initial distance table

Target RTC / Unit RTC	T1	T2	T3	T4	T5
T1	0	0.25	0.75	0.93	0.27
T2	0.62	0	0.37	0.95	1
T3	0.5	0.88	0	0.75	1
T4	0.87	0.56	1	0	0.58
T5	0.5	0.75	1	0.66	0

Table 2(b) The optimized distance table

As provided above in equation 6, the concatenative cost function is a linear combination of the seven factors. Yet, it has been discovered some of them may have interactions. However, to limit the number of free parameters, the numbers of categories for the six textual factors can be reduced, if desired. In the discussion provided below, the number of categories for PinP and PinW have been reduced to 2, while LPhC have been reduced to 4 and RPhC, LTC and RTC have been reduced to 3, although this should not be considered necessary or limiting. In the exemplary optimization method discussed herein, six second-order combinations (i.e. combinations of two textual factors) are investigated at step 1104, in which the maximum number of free parameters is 36; however it should be understood other combinations and/or even higher order combinations (combinations of three or more textual factors) can also be used. In the present discussion,

the combination between LPhC and other factors has not been adopted since these combinations may cause too many free parameters.

As with the single-order components of the cost function, the higher-order components also take only categorical values, the distance between categories are empirically predefined in distance tables. After optimizing the distance tables for these second-order components individually in a manner similar to that discussed above with the single-order textual factors in step 1104, their correlation coefficients to MOS are listed in Table 3. Comparing Table 3 to Table 1, it can be seen that all combinations of Table 3 have a higher correlation than using PinP and PinW alone, yet, only coefficients for LTC-PinW and LTC-PinW pairs are higher than those of using LTC alone. It appears that some of the second-order components play important roles for unit selection. In a further embodiment discussed below, all of the higher-order components are used to form the concatenative cost function at first and some of them are then removed after optimizing the weights since they receive small weights.

	RPhC	*LTC*	*RTC*
PinP	*0.719*	*0.752*	*0.710*
PinW	*0.751*	*0.790*	*0.745*

Table 3

An enumerated smoothness cost is used in the original cost function. Various smoothness costs based on the combinations with the six textual factors have been investigated. In one embodiment, it has been found beneficial to assign the smoothness cost by considering PinW (2 categories), LTC (3 categories) and the final type of current unit (3 categories). That is to say, when the current unit is a continuous segment of its previous segment in the unit inventory, its smoothness cost is set to be zero, otherwise, it is to be assigned a value from a table of 18(=2*3*3) possibilities according the conditions described above. The values in the smoothness cost table can be optimized by maximizing the correlation between smoothness cost and MOS in the training sets, e.g. four training sets. After optimization, the correlation coefficient reaches 0.883, which is higher than the old one, 0.846. This reveals that the new smoothness cost is more suitable than the original one and is used to replace the original one in the cost function discussed below.

Since it is not generally known which component is more important, at first, the new cost function in step 806 is formed by weighted sum of all the single-order components and higher-order components as discussed above. The weights for each of the components are then optimized as discussed above. An example of optimized weights for 13 components is provided in Table 4. Since some of the components received very small weights, they can be removed from the cost function without much effect.

In a further embodiment, step 808 includes removing some components below a selected threshold and keeping only the more significant components (identified with stars), wherein the weights of the remaining components are optimized again. The new optimized weights in the final concatenative cost function for seven components are given in Table 5. The correlation coefficient between the final cost and MOS reaches 0.897, which is much higher than the original one, 0.822. Speech synthesized with the new cost function should sound more natural than that generated with the original one.

Component	*Weight*	*Component*	*Weight*
PinP	*0.008*	*RPhC–PinP pair*	*0.008*
PinW	*0.008*	*RPhC–PinW pair*	*0.023*
*LPhC **	*0.099*	*LTC–PinP pair**	*0.088*
*RPhC **	*0.054*	*LTC–PinW pair**	*0.113*
*LTC **	*0.104*	*RTC–PinP pair*	*0.008*
*RTC **	*0.091*	*RTC–PinW pair*	*0.016*
		*New smooth cost **	*0.380*

Table 4

Component	*Weight*	*Component*	*Weight*
LPhC	*0.061*	*LTC–PinP pair*	*0.122*
RPhC	*0.059*	*LTC–PinW pair*	*0.170*
LTC	*0.016*	*New smooth cost*	*0.481*
RTC	*0.091*		

Table 5

At this point it should be noted the utterances used for the MOS experiment should be designed carefully so that units have wide coverage for textual factors. In the example discussed above,

prosodic feature orientated CART indices have been adopted for all units, where most of the units used in the MOS evaluation take zero costs for their PinP and PinW factors. Thus, the two factors show smaller correlations to MOS, though they can be important factors. On the other hand, optimization using 400 utterances is not enough for training all the parameters. If possible, a larger scale MOS evaluation can be used to get more reliable optimized parameters. Since the result of MOS evaluation can be used perpetually, (i.e. over and over), it may be worthwhile to do a well-designed large-scale MOS evaluation.

Although the present invention has been described with reference to particular embodiments, workers skilled in the art will recognize that changes may be made in form and detail without departing from the spirit and scope of the invention. In particular, although context vectors are discussed above, other representations of the context information sets may be used within the scope of the present invention.